Archimedes

TWIN

USER GUIDE

TWIN

User Guide

Archimedes

All correspondence should be addressed to:

Customer Support and Training,
Acorn Computers Limited,
Cambridge Technopark,
645 Newmarket Road,
Cambridge CB5 8PD.

First published 1987
Issue 1 1987
Published by Acorn Computers Limited
Part number 0481,163

Contents

1. Introduction

Twin is a windowing editor specifically designed for use on ARM-based machines. Particular attention has been paid to its ease of use and interaction with assemblers, compilers and high-level languages, all of which can be run directly from Twin.

Compilation errors, reported by their line number, can be diverted to Twin files and shown on one of the screen windows. The other screen window can then be used to display the offending line of source, and the programmer can make any required alterations, deletions or insertions in the text. Work on the text may continue while some other task is running concurrently.

1.1 Facilities

- Edits text from one of ten internal memory buffers.
- Views portions from two buffers simultaneously.
- Runs background tasks, the results of which can be seen in a screen window.
- Copies text from one window to another.
- Transfers copied text as an input to a background task.
- Goes to specific lines in the text.
- Quickly finds, replaces or counts text specified in a variety of ways.
- Provides on-screen help and status information, if required.
- Uses filing system and operating system commands.
- Maintains a software clock and date and time-stamps document files.
- Prints text.

When Twin is first loaded, a window is set to the same line and column dimensions as the screen, and it takes its text from buffer 0. The window size can be reduced, enabling the screen to display both static and dynamic help information. Alternatively, a second window which takes its text from a different buffer may be called on to the screen. Twin can organise as many as 10 memory buffers, of which any two can be viewed by screen windows at any given time.

1.2 How this guide is arranged

The guide has five main chapters. This introduction shows how to enter and leave Twin and how the screen is arranged. The second chapter describes the different sorts of file you can edit, and how to use Twin concurrently with other programs. The third chapter covers how to edit files, and the fourth describes Twin's powerful find-and-replace command. The final main chapter describes how to save, load and combine files.

At the back of the guide, a series of appendices give some useful reference information. You'll find Appendix A, which summarises Twin's function-key commands, particularly useful.

1.3 Conventions used in this manual

Text printed like this is text that you type or see on the screen:

`colour`

Keys you press are shown like this:

[Return]

When you need to press two keys together, we show the instruction like this:

[Shift]-[F1]

A set of standard names is used for keys, though they may be labelled differently on your computer. The standard names used are:

- [Esc] for the escape key
- [Tab] for the tab key
- [Ctrl] for the control key
- [Shift] for the shift key
- [Delete] for the delete key
- [Return] for the carriage return key.

Text printed like this shows the sort of item to type:

filename

Here you would type a filename, not the word filename.

{, numeric}

This example shows that you may type a comma then a numeric. The curly brackets indicate optional items in a syntax line.

To give a Twin command, you press a function key. Each key can give up to four different commands, depending on whether you press it on its own, with [Shift], with [Ctrl] or with both the shift and control keys. In this guide, we show these commands as command names in boxes, rather than as particular numbered function keys. For example, pressing function key one is shown as:

[command line]

Pressing [Ctrl] and function key three together is shown as:

[next line]

In addition, each function key can have two further meanings, available when you have pressed [find and replace] or [global replace]. All the function key commands are shown on the function keycard, supplied. We suggest you insert it in the function keycard holder at the top of the keyboard before using Twin.

1.4 Starting Twin

Twin is supplied on a 3.5-inch ADFS D format floppy disc.

Twin may be loaded in three ways from the Arthur supervisor prompt:

`Twin` [Return]

creates a blank document in buffer 0

`Twin` *filename* [Return]

loads Twin and the named file into buffer 0

`Twin` *filename filename* [Return]

loads Twin, the first file into buffer 0, and the second file into buffer 1. Both windows are displayed.

Any file may be loaded into Twin; it need not be an ASCII file.

1.5 The screen layout

When Twin is entered by typing:

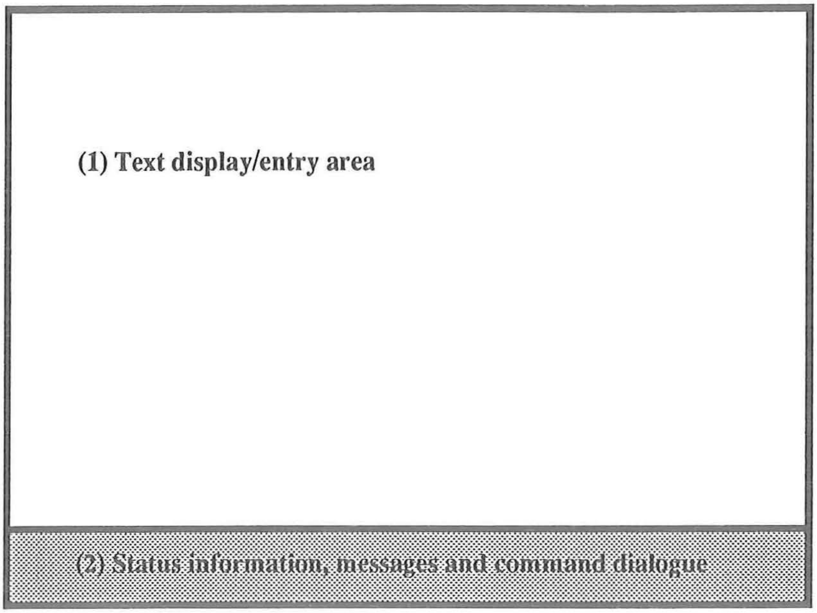
`Twin` [Return]

or:

`Twin` *filename* [Return]

only buffer 0 is active and one screen window is displayed.

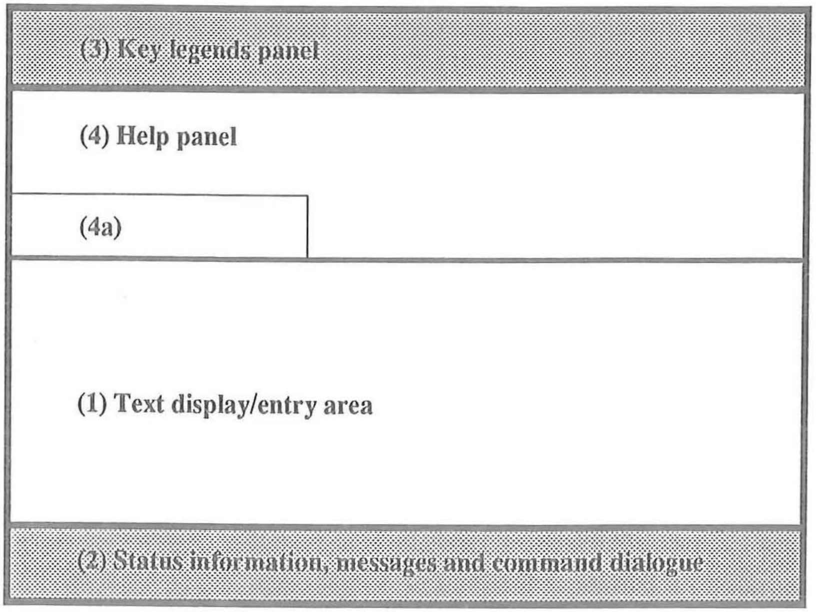
The screen format is as shown below:


(1) Text display/entry area
(2) Status information, messages and command dialogue

This screen layout is only one of several available. Until you are familiar with Twin, the descriptive mode screen will probably prove the most helpful. This mode is selected by pressing:

[set mode] D

There are now four areas on-screen:


(3) Key legends panel
(4) Help panel
(4a)
(1) Text display/entry area
(2) Status information, messages and command dialogue

Text display/entry area

Displays a portion of the text in the buffer, including the text being typed. This area may contain one or two windows, which need not necessarily be identical in size.

Status information, messages and command dialogue

Usually indicates the current text entry mode and number of marks set, but is also used for information, warning and error messages, command prompts and replies to prompts. The status line has six fields:

example field	*explanation*
`01`	left digit: buffer number of current file right digit: number of pushed buffers (or blank)
`Insert`	characters will be inserted into the text
`Over`	characters will overtype the existing characters
`Tab cols`	moves the cursor across the screen or the current line by fixed column spacing of eight characters
`Tab words`	moves the cursor along the current line to positions directly below that of the first letter of each word on the line above
`One mark`	one mark has been set into the text
`Two marks`	two marks have been set into the text
`Original`	the current text has not been modified since last saved, or is a new document
`Modified`	the text has been altered since it was loaded or saved
`Discarded`	the text has been thrown away after replying Y to the safety prompt
`LF <-> CR`	original text, except that linefeed and carriage-return have been interchanged
" *filename* "	the name of the current file, plus a 3 character file type if the file is dated and stamped, and is not FFF (ie not a text file).
date	the date of a dated file

`**Command File**`	a file which has a load address of 0 and an execution address of FFFFFFFF and which will be read as a series of commands by a filing system
address>	the load and execute addresses of a binary file.

Miscellaneous information, such as the results of global replacements, is also shown in this line.

Key legends panel

Describes the Twin command provided by each function key.

Help panel

By default this area describes the functions of the cursor keys. In addition to this information. The bottom left-hand corner of this area (marked 4a in the diagram) describes the action of the Twin commands when they are selected.

1.5.1 Increasing the text area

The text display area may be increased at the expense of the help messages by selecting the key legends (K) mode. This is achieved by:

set mode `K`

The key legends may be removed with:

set mode `3` - to give 25 lines of text, or
set mode `0` - giving 32 lines of text

The D and K modes provide helpful information about the editor, but have the disadvantage that they leave less room on the screen for text than display modes 0 and 3.

1.5.2 How special characters are shown

Control codes are displayed as their corresponding letter highlighted. For example, Ctrl `A` is shown as an inverse video A.

Linefeed characters are normally hidden from view unless they need to be shown. To display linefeeds, press:

display newline

Linefeeds appear as an inverse video left-arrow. The carriage-return character (CTRL M), displayed when you press change LF<->CR cannot be hidden. The end of the buffer marker is displayed as an inverse video asterisk.

The top status line

If you value an on-screen time display, but do not want to use mode D, there is mode T, obtained by pressing:

set mode `T`

This gives a status line similar to the following abbreviated version:

```
Twin'T'mode  Shift  F5  to  change  mode  Twin  at  &00060000
10:16:46 09-Jul-87
```

Amongst other things it shows the time and date, and also the base address of the Twin in use, (ie HIMEM for any tasks run with Twin).

1.6 The keyboard

The letter, number and punctuation keys have their normal functions within Twin, but some of the others have modified functions:

key	*action*
Return	types new-line character (ASCII &0A), moves cursor or completes a command
Tab	moves the cursor (2 modes)
Shift-Tab	switches tab modes
Ctrl-Tab	expands tabs in text
F1 to F10	invoke Twin commands
Shift-F1 to F10	invoke Twin commands
Ctrl-F1 to F10	invoke Twin commands

key	*action*
Ctrl-Shift - F1 to F10	invoke Twin commands
Copy	deletes the character at the cursor
Shift-Copy	switches cursor to cursor copying mode
Ctrl-Copy	deletes the line marked by the cursor
cursor keys	moves cursor a character/line at a time
Shift with left or right cursor	moves cursor left or right by a word
Shift + up/down cursor	moves cursor up/down by a window-page
Ctrl with left or right cursor	moves cursor to the start or end of line
Ctrl with up or down cursor	moves cursor to beginning or end of text
Ctrl-Shift + up/down cursor	moves both windows up/down by a page
Esc	cancels an incomplete command or leaves cursor editing mode.

The function keys are specially defined by Twin. Their functions will be explained as and when necessary. A brief description of all of them is given in Appendix A.

1.7 Leaving Twin

To go back to the operating system, press:

exit to language   Return

You will be prompted to specify the language to which you want to return. Pressing Return by itself takes you back to the operating system.

2. Working with Twin

2.1 Types of file

Twin can create three types of files:

(1) **Documents**, which may be any text or language-related source-code. A document is a file which has been date-stamped automatically by Twin. The current date and time is obtained from the Archimedes' real-time clock. Should the time not be set, Twin uses the **Command file** form of document, explained below

(2) **Command files**, which can be used in the same way as a document or date-stamped file, but they are marked as having a load address of -1 and an execution address of 0, enabling the operating system to use them as a series of input instructions. Command Files can therefore be executed under the `Twin*` or Arthur supervisor prompts and will perform automatic actions

(3) **Pure binary files**. The file may be inspected, and it can be edited by overtyping. Usually, it would only be sensible to edit recognisable ASCII strings. If edited in insert mode, the file would be increased or decreased in length with fatal consequences upon branch instructions. The load and execution addresses of the loaded binary file are shown on the status line.

Twin can edit all other typed files and shows the type as 3 hex digits to the right of the file name.

2.2 Twin and the operating system

Any of the filing and operating system commands may be used from the editor, by first pressing:

[command line]

The bottom of the screen displays the `Twin*` prompt, and the operating system can be used. There are two options to return to Twin:

[Esc]

or

[Return]

when the `Twin*` command line is blank.

2.2.1 Command files

Although Twin is normally used directly by the keyboard operator, it can also be driven by command files which have been created for the purpose.

Command files are best used when you wish to repeat the same editing operations on several files or when the editing is such a complex sequence that it is best done in stages.

Since command files execute without your interaction, their construction and testing should be done in stages so that careful checks can be made to ensure that a command file does what it is supposed to do and no more.

The first line of a command file should be blank.

The following example sequence of commands makes a language the background task in a Twin-window environment.

Enter Twin in command file mode and type:

[Return] [enter character] [toggle window] [enter character] [push buffer] [enter character] [task status] *language name* [Return] [enter character] [toggle window]

This enters six characters (plus the number of characters in the language name) into the Twin command file. These commands are typed into Twin using the special [enter character] function. For example, [enter character] then [toggle window] places the toggle window character into the command file, and so on. The screen will show whatever symbols are associated with these characters: they are likely to be blank, or an assortment of graphic, mathematic and foreign language symbols.

Note the use of [push buffer] to ensure that the language file doesn't overwrite any Twin file when it is loaded.

Such command files don't always have to run under Twin, as shown in the example given above. They always start in the operating-system environment, but may call, enter and leave language environments. Twin will often be involved because the command sequence is likely to want to make use of Twin's editing features. This type of command file is often used as a filing system !BOOT file, to select the editor and set options (such as display carriage returns, release scroll margins, select text colour, obtain date and time from a network and so on) at the start of editing.

Text in Twin can be placed into a language by pressing [exit to language] then the language name. After checking to see if any altered text needs to be saved to the filing system, Twin will goes to the named language. Alternatively, you can go to the operating system by pressing [exit to language] then [Return].

Loading and saving with the filing system is described in chapter 5.

2.3 Twin concurrency

To run a task concurrently with Twin, it is necessary to select a buffer to run it in. Pressing [task status] displays a command prompt on the new window's status line, and a task can be called – it may be a language or a utility. All output from the task prints to the buffer for as long as the task runs, and the contents of the buffer can be viewed in the Twin window. Editing on another of Twin's buffers can take place while the task is running, but the task will stop occasionally as Twin loads and saves files, or gives prompt and error messages which require some immediate response from you. Twin's clock will not be updated.

When the task's buffer is on-screen, the window is the output screen for the task, and keyboard information (characters typed in the range &00 to &7F) can be passed to the program. All the editing and cut-and-paste facilities of Twin are retained, and are available to assist in entering commands into the task program. A [marked copy] operation can be performed between the Twin window and the task window which directs all the copied characters to the task. The task window normally shows the information flowing into the end of the task buffer – in this way the latest output is always on view. This is a task-linked situation, the default. However, pressing [auto bottom] unlinks the task so that the task window no longer follows the data streaming into the buffer. The task-unlinked state allows you to move the cursor all over the task buffer.

[auto bottom] toggles between the task-linked and task-unlinked state. The task window can be re-sized, and may be removed from the screen altogether. If you are in a Twin window and it is inconvenient to go into the task window, pressing [task bottom] forces the task-buffer cursor and window to the end of the task buffer.

When a task is started, half of the available memory is allocated to it for the output: this will be used up gradually, but there is no check for running out of memory. Only one task may run at one time, and a new task cannot be started until the one running has finished or is cancelled.

Twin can read the error messages from compiler output and go to the offending lines in the source file. With the source code in your current window, and the compiler output in the other window, press [next·line].

Twin now looks for the first occurrence of `line` (in any case of characters) in the compiler output. For example, it might find:

```
Error in line 98
```

The cursor now moves straight to that line in the source file. In this example, it moves to line 98. You can now correct the error in line 98 and repeat the process to find the next line which the compiler has identified as containing an error.

In the task-linked mode of operation, Twin must redraw its window to display any information arriving from the task: this slows down the task in hand. If faster operating speeds are required, the task window may either be reduced in size to its smallest dimension (five lines), or turned off completely. Repeated pressing [expand window] while in the Twin window (as opposed to the task window) reduces the size of the task window, and pressing [close window] while in the task window removes the task window from the screen.

A simple worked example of how to run the ARM assembler from Twin is given in Appendix B.

3. Editing

3.1 Insert and overtype modes

Twin offers the choice of typing in overtype or insert mode. In overtype mode, the characters typed replace those currently at the cursor position on the screen. In insert mode, they push existing text to the right to make room for themselves. The mode in use is displayed in the bottom-left corner of the screen (the status area). To switch between these modes, use insert/over . Certain keys behave differently in the two modes – these differences will be mentioned as the functions of those keys are introduced.

When the text you are typing reaches the right-hand side of the screen, it overflows on to the next screen line; no special character (such as hard or soft returns) is placed in the text at the point where the text breaks to start a fresh line on the screen. Therefore, to start a new line, it is necessary to press Return . Twin imposes no limit on the number of characters that can be typed on one line, other than the maximum buffer size, dictated by the available memory.

Pressing Return in insert mode puts a new-line character (ASCII 10) at the current cursor position and the cursor moves to the start of the next screen line. The return character is important as it marks the end of a line to the editor. Return characters are normally invisible. To make Twin display return characters, press display newline . Return characters show as left-arrows on a highlighted square.

Pressing Return in overtype mode moves the cursor to the start of the next screen line.

To correct minor errors while typing, use Delete to erase characters to the left of the cursor. In insert mode, this removes the character from the screen and buffer; in overtype mode it doesn't remove it, but replaces it with a space.

3.2 Moving around the text

The cursor keys have a repeat action if they are held down. The screen window can only show part of the program text; this is regarded as a page. To see another page, use the cursor keys as below:

- to move through the Twin buffer a line at a time (scrolling), use the up or down cursor keys – the screen will scroll when the cursor comes within a few lines of its top or bottom (this distance is the scroll margin, and can be changed using [change margins])
- to move through the text buffer a page at a time, use the up or down cursor keys with [Shift]
- to move to the start or end of the text buffer, use the up and down cursor keys with [Ctrl]
- to move the cursor under the start of words on the preceding line, press [Shift] - [Tab] until `TAB words` is displayed on the status line at the bottom of the screen, then use [Tab]
- to move the cursor to the next tab position to the right, press [Shift] - [Tab] until `TAB cols` is displayed on the status line at the bottom of the screen, then use [Tab]
- to move to a particular line in the buffer, use [go to line] . This displays the current line number (`At line` *nnn* , `character` *xxx* , `new line:`), where *nnn* is the number of returns between it and the start of the buffer. You may now type a line number, or + then a number to move forwards that number of lines, or – then a number to move backwards
- to move to a known piece of text in the buffer, use [find and replace] (see chapter 4.1).

The scroll margins

The screen will normally scroll when the cursor is moved to within four lines of the top or bottom of the screen (the scroll margin) to ensure that text surrounding the cursor can be seen properly. It is possible to alter the action of scrolling by the [change margins] key. Using [change margins] displays the prompt:

```
set Bottom margin, set Top margin or Remove margins ?
[B/R/T]
```

[change margins] B

sets the bottom margin to the cursor position

[change margins] T

sets the top margin to the cursor position

[change margins] R

removes the scroll margins altogether.

The margins will be reset to Twin's preferred positions if the size of the window is altered by pressing [expand window] [toggle window] [close window] or changing screen modes. Toggle window operations which do not reset the size of the windows do not reset the margins.

3.3 Changing the text

3.3.1 Small alterations

Small changes to text in the buffer are made using the [Delete] or [Copy] keys for single-character deletions backwards and forwards, or by overtyping in overtype mode.

In insert mode, [Delete] deletes characters to the left of the cursor, [Copy] deletes characters to the right of the cursor, and typing inserts the new text at the cursor.

In overtype mode, [Delete] replaces characters to the left of the cursor with spaces and moves the cursor to the left, [Copy] behaves as it does in insert mode, and typing replaces existing text with new text.

To remove a blank line or join two lines, the new-line character, together with any spurious spaces which might be present, should be removed. This is often easiest to do when the new-line character is made visible by pressing [display newline].

Whole lines may be removed by pressing [Ctrl]-[Copy] simultaneously.

3.3.2 Cursor editing

To copy short sequences of text from the screen, place the cursor at the position where the copied text is required and press [Shift]-[Copy] to select cursor editing mode. Next, move the cursor to the text which is to be copied. Pressing [Copy] now copies the text character by character. Extra text may be typed in and existing text may be deleted using the [Delete] key. The copying process cannot continue past a new-line character. Press [Esc] to return to insert or overtype mode.

Two drawbacks to cursor editing limit its usefulness: control characters (including the new-line character) cannot be entered as part of the copy, and, if the destination line is before the source line, copying may invoke a line-scroll and so impair a successful copy from that point onwards.

The singular advantage of cursor editing is that it disables the editor commands on keys [F1] to [F10] so that you can use those keys to type complete words, phrases or ASCII codes 127-255 with a single keystroke.

Although these techniques are very useful, even more powerful editing commands are provided to delete, move and copy blocks of text.

3.3.3 Cutting and pasting text

These blocks are only limited in size by the text in the buffer. To use any of these commands, you must first define the block of text to be used. This is done by moving the cursor to the start and/or end of the block and marking its position with [mark place].

To delete a block of text, mark the start or end, then move the cursor to the other end of the block and use [marked delete].

To move a block of text, mark the start and the end (in either order), then move the cursor to the destination outside the block and use [marked copy].

To copy a block of text, mark the start and the end (in either order), then move the cursor to the destination and use [mark place]. From now on, each time [marked copy] is pressed, a further copy will be placed at the cursor position, until the marks are erased using [clear marks].

To copy a file from the filing system into existing text in the Twin buffer, position the cursor where the text is to go, then use [insert file], typing the file name in response to the prompt `Type filename to insert:` Twin removes marks automatically after the block delete or move commands. Marks are not removed automatically after the copy command, to allow multiple copies of marked text to be made easily. Marks can be cleared at any time with [clear marks]. It is best to do this as a matter of good practice, as unwanted marks can modify the action of certain commands such as [global replace] and [save file], and can prevent operation of some commands, causing the error message `Bad marking` .

The number of marks currently set is displayed in the status line at the bottom of the screen. A maximum of two marks is allowed in any of Twin's buffers: buffers have their own set of marks. This allows copying between one buffer and another, providing that marks have first been cleared in the destination buffer. The text marked should be in the source buffer, and the cursor placed in the destination-position of the current buffer. A copy (but not a move) is then made between the source and destination buffers.

Twin will use as its source buffer either the buffer which occupies a screen window, or, in cases where only the destination window is on the screen, the buffer which has just been removed from the screen using the close window key.

3.4 Clearing and restoring text

To delete the entire contents of the text buffer, use clear text , then use one of the options in response to the following prompt:

```
Clear text Y (dated),shf-F9 (auto-exec),D {discard}/N
```

`Y` kills the current file and begins a fresh one which will be date-stamped.

clear text kills the current file and begins a fresh one which will be executable as a command file.

`D` removes the altered status from the current file. The file remains in memory but it will now be possible to remove it by a load file command which does not ask permission before overwriting the file with the new file.

`N` means do not clear text. Esc may also be pressed, in which case the text is not cleared. If no action is taken, the command times-out and automatic escape occurs.

Use clear text very carefully. There is no command to restore lost text.

4. Finding and replacing text

Twin provides two very powerful search-and-replace commands, which can perform the following types of functions:

- rapidly changing all or selected occurrences in the Twin buffer of one string into another string – correcting spelling mistakes, improving consistency, expanding or shortening variable names in programs, expanding abbreviations
- finding the next time a text string appears in the Twin buffer – moving to a line where the content is known, but the line number is not, checking procedure/function parameters in programs and so on
- counting the number of times a string occurs in the Twin buffer – checking the size of a document, guarding against over use of particular phrases, analysing style, looking for under-used variables, procedures and functions in programs and so on.

The two commands are:

- [find and replace], which you use to look for or replace selectively a phrase from the current cursor position to the end of the file
- [global replace], which you use to look for or replace globally a phrase throughout the whole file, or within a block in that file; or to count the number of times a phrase occurs in the file.

You use both commands in the same way. After pressing the function key, you type the text you want to look for – the target string. The function keys have now taken on a new set of meanings, shown in white type on the key card. These keys let you specify the target string in very precise ways.

Having entered the target string, you then enter the string with which you wish to replace the target string - the replacement string. You do this by first pressing [by] and then entering the replacement string. When you are happy with the target and replacement strings, press [Return].

To search and replace within a block, move to one end of the block and press [mark place], then move to the other end and press [global replace].

4.1 Simple finding and replacing

To find a phrase, press:

[find and replace]

The screen displays the prompt `Find and replace:`
Type the target string, followed by [Return] . Twin now looks for the string, ignoring the case you used to type it. For example, if you specify `book` , Twin looks for `book` , `BOOK` , `BoOk` and so on. If it finds it, it shows the prompt:

```
C(ontinue), E(nd of file replace), R(eplace), L(ast
Replace) or ESCAPE/RETURN
```

Pressing `C` causes Twin to look for the next occurrence of the string.

Pressing `E` causes Twin to prompt for a string to replace the target string from the cursor position to the end of the file. If you press [Return] without a replacement string, the target string will be deleted from the cursor position to the end of the file.

Pressing `R` causes Twin to prompt for a string to replace the target string at the cursor position. If none is entered, it will delete the target string at the cursor position and then move on to the next string.

Pressing L causes Twin to prompt for a string to replace the target string at the cursor position. If none is entered, the target string is deleted. No further searches are undertaken.

You can carry out the same find and replace operation again by pressing [find and replace] [Copy] [Return] . The [Copy] key recalls the target string you last looked for. [shift] [Copy] recalls the string last looked for with the global replace command.

4.2 Globally deleting

To delete a string throughout a document, press [global replace] . You will be prompted for the string you want to remove. Enter the string, followed by [by] [Return] . You will be asked to confirm you want the text deleted and nothing put in its place. Type a Y or press [global replace] again to execute the command.

Note that pressing [global replace] [Return] will automatically execute the global replace command you last set up, and may not have carried out. If you want to abort a global replace, press [escape].

4.3 Counting occurrences

To find out how often a phrase occurs in your file, press:

[global replace]

The screen displays the prompt `Global replace:`
Type the target string, followed by [Return]. Twin now counts the occurrences of the string.

4.4 Specifying a target string

This section describes the more sophisticated ways in which, using either [find and replace] or [global replace], you can specify the text you want Twin to look for. To do this, you use the special function key commands that become available after you have pressed [find and replace] or [global replace].

To find:

- a phrase, the first character of which is in a specified case, press [exact] then the phrase. For example, to find PC or Pc, but not pc, use [find and replace] [exact] PC. This feature is more often used for specifying the replacement string (see 4.4 below).
- a phrase, part of which must be in a specified case, use [case]. [case] switches on case sensitivity, and switches it off if pressed again. This is useful when you want to find a phrase where certain parts of it must be in a specified case. For example, `acorn` [case] `Computers` [case] `limited` would find `Acorn Computers Limited` and `ACORN Computers LIMITED`, but not `ACORN COMPUTERS LIMITED`. Toggling on [case] alerts Twin to look for Computers with an initial capital letter and the rest lower case. It will reject all other variants.
- a new-line character, press [new line]
- a control character, press [control] then the appropriate character

- ASCII 128-255, press [set top bit] then [control] and the appropriate character – for example, for ASCII 129, you'd press [set top bit] then [control] A
- ASCII 257, use [set top bit] A
- any single letter, digit, or underscore, use [alpha-numeric]
- any single digit (0 to 9), use [number]
- any single character (ASCII 0-255), use [any character]
- any one of a set of characters, for example, xyz, press [start set] xyz [end set] . Note that this command is case-sensitive: in this example, Twin would find y but not Y
- any one of a range of characters, eg, for f to i, press f [to] i . This command is also case-sensitive: in this example, Twin would find h but not H
- any character other than that specified, use [not item] – for example, to find anything other than n, type [not item] n.

You can use [start set] , [end set] , [to] and [not item] with other single-character specifiers, so that the following examples are all valid specifications of single characters:

[start set] [alpha-numeric] $ [end set]

matches any alpha-numeric character or the dollar sign

[start set] @a [to] z [end set]

matches any lower-case letter or @

[not item] [start set] [number] [end set]

matches any non-numeric character

[not item] 3 [to] 6

matches any character other than 3, 4, 5 and 6

[start set] a [to] ce [to] z [end set]

matches any lower-case letter other than d.

4.4.1 Specifying variable-length target strings

Two extra function key commands let you specify strings that don't have a fixed length:

most items

many items .

Most items

This command lets you search for strings made up of one or more of the same character. For example:

most items `1` finds any occurrence of `1` , `11` , `111` , and so on

most items alpha-numeric finds any word

most items number finds any number (6, 66, 3454545)

most items not item new line matches a non-blank line.

This command is particularly useful for finding multiple spaces.

Many items

This command lets you specify the start and end of a target string, without specifying what happens in between. You normally use it in conjunction with not item , any character , alpha-numeric , or number . For example, to find every phrase that starts `Acorn` and ends with the `L` of `Limited`, you would type:

`Acorn` many items any character `Limited`

Note that this would also find examples of "Acorn" where the "Limited" might be many lines or even many pages further on.

To delete ends-of-lines from a `;` character up to the end of the line, you would search for:

`;` many items any character new line by newline

and replace each occurrence with a newline. This particular target-string specification asks Twin to look for every phrase starting with a semi-colon, then including a series, of unspecified length, containing any characters and ending with a new line character.

The [many items] key is most useful for specifying unimportant filler characters, that is, characters that need not be there at all to match the target string.

4.5 Specifying a replacement string

Straight after typing your target string, you can specify a replacement string by typing [by] then the string.

As before, you can use any of the special search-and-replace function key commands.

You can transfer the target string into the replacement string using the command [found string] . For example, to replace all occurrences of micro with microcomputer the command would be:

[global replace] `micro` [by] [found string] `computer` [Return]

The whole find string `micro` is incorporated into the replace string.

Where you are not able to specify exactly what the found characters will be, for example where they are variables, you can nevertheless drop one or more of them back into the replacement string again unchanged. To do this, you first use a number of ambiguous descriptions such as [alpha-numeric] or [number] to define what you are looking for in the target string, then use [field#] followed by a number, in the replacement string. The number specifies which of the ambiguous target characters you want to use again. For example:

[global replace] IC [alpha-numeric] [alpha-numeric] [alpha-numeric]
[by] IC [field#] 0

means "Look for all instances of IC followed by three numbers or letters and replace them by IC followed by the first one of these numbers and letters". The 0 refers to the first [alpha-numeric] . These ambiguous fields are numbered 0, 1, 2, 3 etc. In the example above, if you wanted the third of the variables to be repeated in the replacement string as well, you would type:

[global replace] IC [alpha-numeric] [alpha-numeric] [alpha-numeric]
[by] [field#] 0 [field#] 2

5. Files

5.1 Saving and loading files

5.1.1 Saving text files

The save file key is used to save a Twin document to the filing system. save file produces a bleep to warn you that a file of the same name in the current directory of the filing system may be about to be overwritten. This is in case save file was selected by mistake. There are a number of ways in which to give a filename to the document being saved:

- type a filename then press Return in response to the prompt `Type filename to save:`
- save the document using the same filename as last typed for save, load or insert by pressing Copy Return after `Type filename to save:` (After pressing Copy but before pressing Return the filename may be altered)
- place > *filename* on the first line in the buffer and then press Return when prompted for the filename. This line must be less than 129 characters long but > *filename* need not be at the start of the line.

To save just part of the buffer in a file, use mark place to mark one end of the part required, move the cursor to the other end and then use save file . The prompt:

```
MARK TO CURSOR save:
```

will be given instead of:

```
Type filename to save:
```

No bleep will be given.

5.1.2 Loading text and program files

The contents of the currently selected buffer may be replaced at any time using [load file]. The prompt:

```
Type filename to load:
```

will appear and the filename followed by [Return] should be typed.

The file named should be an existing file on the current filing system. Any type of file currently supported by the filing system or language will load into Twin. The wildcard * character may be used where the filing system allows: for example, [load file] `P*` instead of [load file] `Paintpr4`. To ease the typing of load names which are complex due to directory routing, the special [load via * path] command is available (see below).

Some types of file, such as ARM object code, will load correctly but will not be very readable, as they do not consist of many ASCII strings. They can be edited, however. Some programming languages convert keywords in their programs into ASCII codes above 127. Programming languages may provide commands to pass their tokenised programs to the editor, expanding them to readable plain text in the process: for example, Archimedes BASIC. Refer to the language's documentation for guidance.

A file can also be loaded using the first and second methods outlined in section 5.1.1.

5.1.3 Combining files

Files can be added to the text in the buffer rather than completely replacing it by using [insert file]. The file is inserted at the cursor position.

5.1.4 The load via * path facility

When you regularly have to load files a long way from your current directory, Twin has a feature which saves you having to type in a long pathname. This is [load via path]. You create a file called `path` in your current directory, and list the full pathname(s) of the file(s) which you will want to load, starting at the root directory. For example, where you have three directories which contain files you may want to load, you would enter the following in the file `path`: [Return]

`$.arm.library` [Return]
`$.arm.peter.utils` [Return]
`$.armdoc.memos` [Return]

This example assumes one file has already been saved to `$.armdoc.current`, another to `$.arm.peter.utils` and a third to `$.armdoc.current`. The first line is blank thus also allowing Twin to look in the currently selected directory.

Once the path file has been set up, press [load via * path] and enter the filename in response to the prompt

```
Type filename to load (via *path):
```

Twin will look for the file's pathname in `path`. When it has found it, it will go to that directory and load the file as requested. When the file has been found, it will be displayed and the full path name will be placed in the file name position on the status line.

Names and delimiters can be built up as required. Use only |, ASCII 10 or ASCII 13 as delimiters. Using | (which can be read as or), the example path file would look like this:

```
|$.arm.library|$.arm.peter.utils|$.armdoc.current
```

There is still the null name before the delimiter, so the current directory would be searched first. The directory names given can be relative to the current directory. The path file must not be longer than 1024 bytes.

5.2 Printing files

Twin can print the contents of the currently selected buffer directly to a printer and the output will simultaneously appear on the screen, scrolling at a rate determined by the printer's baud rate or the size of the printer's internal buffer or both. If a single mark is in the text, the printed output will consist only of the text between the mark and the cursor. If two marks exist, the output will be the text between the marks.

Twin normally makes no attempt to format lines and consequently words tend to become split at the right-hand edge of screen lines and printed lines. Word-splits tend not to occur very often in source code where the average line length is less than 80 columns, but for sections of explanatory text or for documents, word-splitting is not acceptable. [format paragraph] will act on a paragraph of text and remove any split words by selectively replacing some spaces with a new-line character. The new line is initially placed after the 78th character, but if this is not a suitable point to break the text, a search is made backwards until a space is found. The paragraph is taken as the text between the cursor position and the next new line which is followed by a character, a space, full stop, comma, semi-colon or colon.

Twin cannot right justify the text.

To print text:

- use [format paragraph] to tidy your text paragraphs
- use [mark place] to define a block within the buffer, if you don't want to print the whole buffer
- press [print text] to start printing.

6. Appendix A

6.1 Twin function keys

[command line]

Commands to the computer's operating system can be given following the `Twin*` prompt which appears on the screen: for example, obtain a CATalogue, run BBC BASIC, and so on. Afterwards, re-enter Twin by pressing [Return] .

[insert/over]

Toggles between insert and over modes for the particular buffer in use.

[expand window]

Enlarges the current window by one line, if there is enough room. The other window is reduced by one line. Cursor positions in both windows remain on screen.

[load file]

Loads a text file into the current buffer. If modified text will be erased in the process, Twin asks you to confirm this.

[insert file]

Inserts a text file at the current cursor position.

[close window]

Closes the current window (the text is not harmed) and displays the other window, which may or may not have been on the screen.

[load via * path]

This loads a text file like [load file] but uses a file path such as *m2path*, *cpath*, or *Twinpath* provided by the filing system. The path file must not be longer than 1024 bytes.

[save file]

There are a number of variations:

- save all text
- save text from a mark to the cursor
- save using the current filename
- save using a name supplied in the text.

Twin date-stamps the file and a beep sounds to remind you that save has been pressed. [Esc] cancels the save if it isn't wanted.

[change margins]

Controls the scrolling action of a window:

- `B` to set the bottom scroll margin to cursor line
- `R` to remove the top and bottom scroll margins
- `T` to set the top scroll margin to cursor line.

A mode change or window command resets the margins to their default value.

[next line]

Twin can look for occurrences of `line` or `LINE` in the error messages of a compiler's output and use the information to go to that line in the source file's buffer.

[append to file]

Adds the contents of the current buffer to the end of an existing file.

[find and replace]

Begins a find or a selective find and replace.

[exit to language]

After checking if there is any altered text to be saved, Twin will exit to a named language (or to the operating system if just [Return] is pressed).

[global replace]

Performs an automatic count of occurrences (with no replace) or a non-selective find and replace. The whole text or just a block of text may be specified.

[set mode]

A number of screen modes are available:

- `0` : 32 line screen mode
- `3` : 25 line screen mode
- `D` : Descriptive: shows all keys and details commands
- `K` : Key legends: shows the function keys names
- `T` : Time: shows the time on the top line.

The current screen mode is automaticallly maintained in CMOS RAM, so that Twin always starts up in the mode selected in the previous session.

[enter character]

Inserts the character from the keyboard directly, or allows the character's decimal code to be typed.

[pop buffer]

A pushed buffer will be recovered from its stack. See [push buffer] .

[mark place]

Puts a marker into the text at the cursor position. Each buffer and each pushed buffer (see [push buffer]) may have up to two marks. When you set a mark, the status line shows `One mark` or `Two marks`.

[clear marks]

Clears all place marks in the current buffer.

[format paragraph]

Formats lines to a width of 78 characters or less, starting from the line the cursor is on, and ending at the first new line that is followed by a non-alpha-numeric character.

[push buffer]

Pushes the current contents of the buffer into stack memory. Only buffers currently selected may be pushed, so two (represented by each screen window) may be pushed at any one time. A new file may be loaded into the buffer vacated by the push. The newly loaded file may also be pushed. In this way files can be loaded and viewed without having to open a new buffer to receive them. Twin can push any number of files, subject only to limitations of memory.

[marked copy]

Copies the text between two marks to the cursor. If there are no marks in the current buffer, text is copied from the alternate buffer (provided it has two marks set in it).

[marked move]

Moves the text between two marked places to the current cursor position which should be outside the marked area. The marks are then cleared.

[print text]

Prints out the whole, or marked, text.

[marked delete]

Deletes the text between the cursor and the marked place. The mark is then cleared.

[change LF <-> CR]

Exchanges all carriage return `|M` and line feed `|J` characters currently in the file.

[auto bottom]

Toggles between task-linked and task-unlinked states. With task-linked, whenever the task window is redrawn, the cursor is moved to the end of the buffer. With task-unlinked, no change is made to the cursor position.

[bytes free]

The number of bytes free in Twin's section of memory. It is possible to adjust Twin's load address to obtain more space.

[clear text]

Deletes all text in the buffer.

`task status`

Starts and finishes a single background task. It will also show whether a task is currently running. The output of the task is appended to the end of the current buffer. While it is running, editing can be done on the appended part of the current file, or in the whole part of a file in a different buffer. Keyboard characters are sent to the task.

`task bottom`

Moves the cursor position to the end of the task window.

`go to line`

Twin computes line numbers by counting new-line characters and three different goto jumps are allowed:

- move to a given line-number. For example, `go to line` `765`
- move to a mark by using `m1` or `m2`. For example, `go to line` `m1`
- move by a number of characters by + *n* or – *n* . For example, `go to line` `-50`.

`display newline`

The new lines can be shown as a special character, a highlighted left-arrow, so that they can be seen clearly. Press this key to either reveal or hide the new lines.

`toggle window`

Toggles between windows. If the second window is not already on screen, then the screen is split and the second window is opened. The selected window's status line is highlighted.

`connect buffer`

Connects a window to a different buffer from the one currently using it, by specifying the buffer's number. There are 10 buffers, called 0 to 9. Alternatively, the plus and minus sign (+ and –) will increment or decrement the buffer attached to the current window. Any response other than `0` to `9`, + or – lists the buffer directory on the screen. Buffers cannot be connected while a pushed buffer is present (see `push buffer`).

7. Appendix B

7.1 Concurrency: a worked example

7.1.1 Twin and AAsm, the ARM assembler

Switch on the computer. When the Archimedes supervisor prompt appears, load Twin by typing `Twin`.

Type the following into Twin:

```
; -> test
 ORG &20000
 SWI 1
 =    "Hello World",10,13,0
 SWI 17
 END
```

[Return]

This short piece of text is an AAsm file, now residing in Twin's buffer 0. The source file can be saved using the name taken from the `; ->` line at the top of the file using [save file] [Return].

Next, obtain a second buffer by pressing [toggle window]. A second, blank, window is now on the screen. AAsm can be called from this window by pressing [task status]. This brings the prompt `Command:` on to the status line of buffer 1. If AAsm is not in the currently selected directory or library, the full pathname must now be given. However, if AAsm does reside there, the response should be:

```
*AAsm
```

AAsm will load and in the window of buffer 1 you will see:

```
ARM stand alone Macro Assembler Version x.xx
Entering interactive mode
Action:
```

The status line will show:

```
1 Task running in this window
```

The response to the prompts should be:

```
Action: ASM

Source file name: test

Code file name: testc
```

whereupon the file test will be assembled.

```
Pass 1

Pass 2

Assembly complete

No errors found

Action:
```

Both passes ran, with no errors reported.

The program can now be executed by leaving AAsm:

```
Action: quit
```

This ends the task and leaves buffer 1 as a Twin buffer, rather than a buffer shared by AAsm. The message `Task running in this window` is removed from the status line. To run *testc* from Twin, it is made another Task. Press:
`task status`

In response to the prompt `Command:` type `*testc`. The task will run, printing "Hello World" to the buffer and causing the status line to report briefly:

```
1 Task running in this window
```

before the task ends and control passes back to Twin.

8. Appendix C

8.1 Error messages

Twin always reports detected errors and the command which is in error is not obeyed. The text in the buffer will be left unchanged but the editor may clear marks or move the text cursor to the start of the buffer or screen line.

`Bad mark number`

You have tried to set more than two marks or have not set the right number of marks for use with this command. For example, you are trying to delete a block with 2 or 0 marks set rather than 1. The marks will be cleared.

`Bad number`

You have typed a number outside the allowed range of letters rather than digits, in response to an editor prompt for a number. Note that numbers are always decimal - you cannot use & to specify a hexadecimal number.

`Bad use of stored name`

While using insert file , you have either pressed Return or Copy Return in response to the prompt for the file to insert, instead of supplying a filename.

`Line not found`

You have specified a destination line with go to line which is beyond the end of the file. (To count the number of lines in the buffer, use:

global replace   new line   Return .)

`No infile name found`

While using the load, save, or insert file commands, you have responded to the prompt for filename by pressing Copy or Return without previously typing a filename. If you did type a filename before pressing Return , the file is either missing or an invalid name was used.

`No previous string`

After using the [find and replace] or [global replace] keys, you have responded to the prompt by pressing [Return] but have not previously specified a find-and-replace string for that command. (Note that the string previously used with [find and replace] is available for use with [global replace] or vice versa, by pressing [Shift] - [Copy] .)

`No room`

There is not sufficient room in the text buffer for the file that you have tried to load or insert, or the text buffer is full so that there is no room left for you to type more text. To release more memory, the contents of a buffer can be saved and the buffer closed.

`Insufficient memory to (re)start Twin at this address`

This error message is most likely to occur when you have forgotten that the language in use is running under Twin and you attempt to edit the program seen as an obstacle. To recover from this error, reload the language.

`Relative move out of file`

On a goto command, the forwards or backwards jump specified would take the cursor out of the file.

`No more lines in other window`

[next line] has been pressed, but there are no more error lines specified for Twin to go to.

`Unsplittable line`

A line has been found which has more than 77 characters, none of which are spaces or new-line symbols.

SKP44